DCG
david consulting group
TRUSTED ADVISOR

How can we run waterfall and agile in parallel?

August 2011

Scope of this Report

- Our focus, as always, is to respond to the question at hand. That means making some assumptions:
  - Readers are familiar with the waterfall and agile SDLC's;
  - In this context, agile primarily means Scrum because although we recognize that there are agile practices that are not scrum (e.g. XP), their inclusion or otherwise does not materially affect the findings of this report;
  - We have ignored the debate over the relative merits of the two approaches because the question implies that there is value in both and in both together. In fact, we agree with this assertion in many instances;
  - Many organizations are already in transition from pure waterfall, "We want to move to agile in a reasonable fashion, meaning pilot a few teams and transition the organization slowly over time – so how do we make sure our agile teams are successful in the interim?" but may not be clear on their end state;
  - Agile and waterfall teams could be focused on existing or new applications to be deployed internally or existing or new products to be sold externally. We will refer to both as applications in this report.

Symptoms of the Challenges when agile and waterfall teams are working on the same applications

- Deliverable synchronization: In a particular project or program, agile teams will have numerous delivery points at the end of each sprint where working code is expected. The waterfall teams will traditionally have one – at the end of the project. Teams will have dependencies on code being produced and tested in parallel by other teams.
- Estimating: Delivery scheduling depends on good estimation of the duration of tasks, assignment of resources and ultimately the estimation of task effort. Unfortunately, estimates can be vague in waterfall because contingency is used as a Band-Aid over the open wound of assuming change cannot happen. They can also be vague in agile because they are perceived to be unimportant if change is bound to happen. In both cases, making and communicating re-estimates in the face of change is not done enough.
- Scale: All software projects are easy if they are small. Frankly, small is manageable but big is a challenge for all SDLC's because we are dealing with: Large complex systems, multiple teams, multiple platforms and matrixed resources.
- Documentation: This is a hugely misunderstood issue but it's out there. It is based on the false notion (held by some agile team members too) that agile teams don't do documentation. Waterfall and agile teams will argue over documentation.

- People: Waterfall teams work on the principle of task separation along lines of specialization. Agile teams work on the principle of task sharing regardless of specialization. This is the reason that a meeting to resolve an issue might require two members of an agile team but 10 members of the waterfall teams.
- Configuration Management: The agile and waterfall teams make different assumptions about the status of the code in the configuration management system and the importance (to the agile team) of frequent builds.

Governance

The project governance model should be established first. It is necessary to define *what* decisions will be made, *who* will make them (one person) and *how* they will be made (who has input and what information will support decision making). Consideration of the voices relevant to each decision is important:

- Voice of the customer (Customer or Customer Proxy)
  - Provides business direction
  - Makes business decisions
  - Prioritizes work on functionality
- Voice of the organization (PMO)
  - Analyzes and reports overall data
  - Facilitates BIG picture coordination
  - Clears external blockers
  - Stays out the way of the team
- Voice of the team (SCRUM Master / Coach)
  - Facilitates team activities
  - Clears internal blockages
  - Leads and does

Planning – Supply Chain

Running waterfall and agile in parallel is a supply chain problem. Planning is essential. Difficult as it may be for some of the participants, success in running waterfall and agile in parallel will come from ignoring the SDLC's and viewing the tasks of the different teams as a logistical exercise in making sure that each task for each team has its pre-requisites in place in time and has all the tools and services it needs to produce its deliverables on time. While this may be a truism for all projects, the criticality here is that the nature and importance of the pre-requisites, tools and services are different for waterfall teams than for agile teams. While planning for the differences, it is also important to remember that compromises may be possible and valid. Here are some key areas in which the availability of the right pre-requisites, tools and services need to be explicitly planned with a supply chain perspective:

- Documentation
- Architecture Approvals
- Project Approval Process
- Configuration mgmt & code builds
- Training (how this project is going to work)
- Customer interaction
- Dev. Environments
- Test Environments
- Procurement e.g. test licenses, dev. Servers.
- Use and availability of stubs
- Test strategy
- Change control process
- Shared resources
- Estimating & re-estimating
- Integration points

Further, it is important to plan for reporting of project progress to the CFO and CEO. Classically, waterfall projects contain a lot of “work-in-progress” and are reported as x% complete in respect of the well-defined (they hope) end deliverables. For agile, a more binary approach can be taken to functionality delivered each sprint (it’s done or it isn’t) but % complete is a much looser concept.

Execution – Agile & Waterfall in parallel

Running waterfall and agile in parallel is a supply chain problem. Openness of information flows, constant monitoring and decisive action in response to disruption of the supply chain are all necessary. The PMO must play a critical role here.

The authors of the Agile Manifesto chose the words “self-managing teams” which have come to be interpreted, perhaps intentionally on the part of the original authors, in some places as “teams who can ignore management.” In big projects, no team can have the luxury of ignoring management if management is the voice of the organization and, hence, the other teams on the project. The self-managing principle is important but, certainly in the context of this question, we recommend that the agile teams interpret this as “self-organizing teams.”

"Agile methodologies free the project manager from the drudgery of being a taskmaster, thereby enabling the project manager to focus on being a leader -someone who keeps the spotlight on the vision, who inspires the team, who promotes teamwork and collaboration, who champions the project and removes obstacles to progress.“

The PMO’s job is to analyze and interpret overall measurement data for the purpose of forecasting the overall project outcome. The sprint teams develop, collect and consume measurement data within the team boundary. Self-managing teams require the necessary data; the PMO should be positioned to ensure that agile teams have the right data to be self-managed. When the overall project outcomes are not expected to meet management and customer expectations the PMO needs to facilitate getting things back on track.

Minimally, the only documentation that should be necessary is documentation that has value outside of the life of the project. For example, it is only minimally necessary to document information that is needed for other people to complete their tasks and for future people to maintain the software later.   It is not necessary to document interim steps of the project but it may be desirable to compromise and meet the waterfall teams half way.

Teams need to coordinate during all planning meetings. Especially important are the release planning meetings, where milestones/deliverables are determined by both waterfall and agile project managers and teams. Continued follow-up and adjustments are covered in iteration planning meetings and daily standups. Scrum of Scrums meetings may be required to keep the waterfall teams up to date, or to help program managers or product managers coordinate progress and status across the teams.

The plan should include identification of those points at which agile teams may mock code or stub out missing code rather than wait for waterfall deliveries if, and only if, the implied action to redo later is scheduled. It is often possible for agile teams to develop against old versions of the application where, for example, an API is being extended but the parts needed by the agile team will be unchanged. Last iterations of agile teams must be more devoted to integration than feature delivery.

Delivery – "Waterfall-at-end"

Regrettably, the agile world turns something of a blind eye to the steps necessary to wrap up a project and make it ready for production. Hence, it makes sense to deliver the project using the disciplines and skills that have been built up over time for the waterfall SDLC. In particular, a separate "production" team is recommended to be in place at the Planning stage.

As with this whole project, the key is to have the delivery phase planned at the start and, yes, it is a supply chain problem. A good analogy here is the building of a ship. Many things can be prepared in advance but at the end of the process the ship has to be assembled in a particular order and then all the sub-pieces have to be tested before the ultimate test of a work-up at sea. Integration testing should be the first step of Delivery with Independent Validation and Verification being an option depending on the organizational risk presented by the project.

While the process may be driven largely by the waterfall model at this point, there are some lessons to be learned from agile. In particular, "do the hardest things first." In the context of the delivery phase this means not arriving at the delivery phase without have attempted all the hard parts of the integration testing in real or simulated fashion during the execution phase.

How can we run waterfall and agile in parallel? Think supply chain. Don't have supply chain expertize or experience? Get some.

Sources (longevity of links not guaranteed!):

"Challenges for Agile Developers in a Process-Driven Waterfall Environment" by Vidya Klein, http://www.taos.com/Newsletter/May%2011/Vidya%20Klein_Hybrid%20Approach%20to%20Software%20Development.pdf
"What killed waterfall could kill agile," by Robert Martin, http://cleancoder.posterous.com/what-killed-waterfall-could-kill-agile

"The Agile/Waterfall Cooperative," presented by Michele Sliger, http://www.rallydev.com/documents/AgileWaterfallCoop-Sliger.pdf

"Agile is from Venus and PMOs from Mars," by Tom Cagley, http://www.davidconsultinggroup.com/includes/getarticle.aspx?resourceid=367&pdf=.pdf

"Agile management for software engineering," by David Anderson. Pearson Education Inc., 2004.

http://www.ccpace.com/Resources/documents/AgileProjectManagement.pdf